UCRL-MI-205645

# Attachment 1

# Draft Statement of Work

# Open Source High Performance Production Cluster InfiniBand Infrastructure

## National Nuclear Security Administration

## Advanced Simulation and Computing

## PathForward Program

Version 22

July 13, 2004

# CONTENTS

# 1 INTRODUCTION

The InfiniBand Architecture (IBA) is emerging as a multi-vendor, commodity, open-standard network interconnect which has proven to have the characteristics of a high performance cluster interconnect: high bandwidth; low latency; minimal CPU overhead; and remote direct memory access (RDMA) capabilities. Current InfiniBand technology is capable of delivering over 2,700 MB/s and sub 5µs (micro-second) latencies through the MPI API to user applications running commodity nodes in large-scale clusters.

InfiniBand coupled with HyperTransport or PCI-Express local I/O channels balances the data bandwidths within a cluster and removes the bottlenecks in the high performance inter-processor communication infrastructure necessary for current large-scale clusters and related storage systems. In addition, the InfiniBand hardware specification has a roadmap to increase data rates that track well with future processor, memory, local I/O, and storage system bandwidth performance. Hence, InfiniBand enables balanced cluster architectures to be built today and into the future with completely commodity components.

The recent formation of the OpenIB organization provides a potential open source community infrastructure for future development of an InfiniBand software stack that could become common across the High Performance Computing (HPC), Enterprise and embedded/real time markets. The OpenIB software stack is the intended final destination for the deliverables required by this Statement of Work (SOW).

Several InfiniBand software releases have been made under open source licenses and have been contributed to the OpenIB repository. These releases can provide a basis for a long term InfiniBand open source software solution that would include HPC capabilities, be sustained by the standards and practices of open source, and be accepted into the Linux kernel and Linux distributions.

Currently many of the open source releases have been implemented independently, optimized within a closed development environment, and focused on enterprise computing and data centers. As a result the current structure of these releases are overly complex, difficult to build, and provides very limited portability across different Linux kernels and distributions, and limited multi-vendor support.

The goal of this project is to accelerate the design and implementation improvements necessary for the completion of an Open Source OpenIB solution. The resulting product will be required to meet the scalability, performance, portability, reliability, and manageability requirements of the ASC (Advanced Simulation and Computing) Program during its next phase, and thereby the HPC community as a whole.

In this SOW we describe a set of features to be developed, project milestones, and testing results to be achieved which we believe will produce improvements to the OpenIB software stack to meet the needs of the National Nuclear Security Administration (NNSA) and the ASC Program that will be production ready, high performance, scalable, extensible, and reliable.

The HPC community has found five key areas where significant design and implementation work is needed in order for the open source software solution to meet the stringent requirements for HPC. These areas are prioritized as follows:

- A Unified Open Source OpenIB stack with HPC capabilities
- Scalable InfiniBand diagnostic and management tools
- Scalable system software and MPI middleware
- Latency reduction

- Platform Independence and Portability

An open source InfiniBand (IB) software infrastructure for high performance Linux production clusters that is adoptable by the OpenIB and open source community is required. The following should be the minimal and sufficient set of software needed for a complete solution: 1) complete InfiniBand access layer including host channel adapter (HCA) device drivers, user and kernel space Verbs interface, connection management, resource management; 2) subnet management and subnet administration; and 3) HPC upper layer protocols for MPI, IPoIB, SDP, and Sandia Portals. This software stack should be portable to multiple InfiniBand hardware solutions (HCA's, fat-tree, Clos, and crossbar switches, and gateway devices) from multiple vendors and InfiniBand silicon providers. In addition, this software stack should be designed to provide a base software environment for multiple hardware IB generations including at least 4x, 4x DDR, 12x, 12x DDR and 12x QDR. The upper layer protocols for SRP, kDAPL, uDAPL, iSCSI, iSER, etc. are of interest, must not be precluded from inclusion in the OpenIB stack, but are not within the scope of this SOW at this time.

HPC places stringent requirements on software including high performance and scalability. NNSA large-scale clusters are using 1,024-8,192 processors today, and are expected to employ 10,000's of processors in the near future. Therefore, this OpenIB software development is required to scale to 4,096 node ports implemented with multiple switches interconnected, with copper or fiber cables, in stages to form fat-tree, Clos, near trees, and meshed networks. In addition, it is anticipated that large SMPs may be employed as cluster nodes. In order to maintain communication to computation (B:F ratio) balance, this OpenIB software stack should efficiently utilize multiple planes of interconnects. This includes multiple switch planes, multiple links per HCA and multiple HCA's per SMP cluster node.

## 1.1 Requirement Definitions

Particular paragraphs of the Statement of Work have the following priority designations.

(a) Mandatory requirements are designated as (MR). Mandatory requirements are items that are essential to the University and reflect the minimum qualifications an Offeror must meet in order to have their proposal evaluated further for selection.

(b) Target Requirements are designated as (TR-1, TR-2 and TR-3). Each requirement so labeled deals with features, components, performance characteristics or other properties that is considered a part of the OpenIB software stack, but will not be a determining factor of response compliance. Target requirements are prioritized with a dash number with TR-1 being the most important. Taken together, the aggregate of the MR, and TR-1 requirements form a baseline OpenIB software stack. TR-1 targets are as important to the program as mandatory requirements, but not meeting any particular TR-1 target requirement is insufficient to render a proposal as non-responsive. TR-2 targets are second priority after TR-1 requirements. TR-2 requirements are considered goals that boost a minimal OpenIB software stack, when offered as an aggregate of MR, TR-1 and TR-2 requirements, into the moderately useful category. TR-3 targets are third priority after TR-2 requirements. TR-3 requirements are considered stretch goals that boost a moderately useful system, taken together as an aggregate of MR, TR-1, TR-2 and TR-3 requirements, into the highly useful category. Thus, the ideal OpenIB software stack will meet or exceed all MR, TR-1, TR-2 and TR-3 requirements. Target Requirement responses will be considered as part of the evaluation of Technical Proposal Excellence (see Attachment 2, Proposal Evaluation and Proposal Preparation Instructions).

In addition to the Mandatory Requirements, the Offeror may propose any Target Requirements or other features for the software, and any additional features consistent with the objectives of this project that the Offeror believes will be of benefit to the University. Target Requirements and additional features proposed by the successful Offeror may be included in the resulting Subcontract.

# 2 OPEN INFINIBAND STACK WITH HPC CAPABILITIES

The recent release of several open source InfiniBand software stacks provides a foundation for developing a unified open source InfiniBand stack with HPC capabilities. The design of this stack should focus on meeting the performance, scalability, manageability, reliability, and portability requirements of high performance computing. This unified stack should be implemented with an open software development environment where a community of open source developers can conduct design reviews and contribute to the code base.

The current situation with multiple InfiniBand stacks has lead to incomplete feature sets, higher code management costs, and incompatibilities between InfiniBand software and hardware. A unified effort within the InfiniBand and open source communities to develop an open source solution would allow the HPC, enterprise, grid, and storage industries to leverage software across multiple InfiniBand hardware solutions and computing architectures.

## 2.1 KEY DEFINITIONS

To facilitate the development of an Open Source InfiniBand software stack, this SOW intends to accelerate the design, development and testing of the "OpenIB" software stack (www.openib.org). This stack is intended to include, from top to bottom, a completely Open Source Software stack (including device driver, access layer, protocols, and tools) that meets the Linux community's requirements, and addresses the performance, scalability, robustness, and reliability requirements of large HPC Linux clusters in production environments. This SOW envisions a rigorous testing regime on large clusters to harden and verify the HPC capabilities of the OpenIB software stack.

To that end this SOW defines the OpenIB software stack as the following components:

- Kernel modules – Changes to the Linux kernel needed to support the rest of the OpenIB software stack.
- Device Driver – Linux device driver to function the IB host channel adapter devices on a node.
- IB Access Layer – Low level API that exposes IB verbs, CM, SM and SA functionality in a vendor neutral and standard compliant manner.
- Subnet Manager – Software entity that discovers network topology, initializes the subnet, establishes routes, and provides regular subnet sweeps.
- Connection Management – Service to establish, maintain, and status communication path between remote peers.
- Subnet Administrator – Service to store, track, and status the IB hardware configuration and routing information.
- Upper Layer Protocols (MPI, IPoIB, SDP and Sandia Portals) – APIs for applications to perform IB communications operations
- Diagnostics and management tools – Commands and APIs shall determine the hardware and software state of the IB configuration and manipulate that configuration.
- Tests harness and modules – Software shall automatically test the functionality, performance, reliability, and robustness of the components of the OpenIB software stack.
- Documentation – All provided software shall be documented so that personnel unfamiliar with the IB stack structure may easily install and manage an IB cluster. Documentation shall

be provided so that personnel may make code enhancements and extensions to any software layer (driver, access layer, ULPs, diagnostics) of the IB software stack.

The following terms will be use throughout the remainder of this document:

- Aggregate Link Bandwidth (LAB) is defined as the minimum of the aggregate memory bandwidth, aggregate bus bandwidth, or the sum of uni-directional link peak user payload data bandwidth.
- Aggregate Bi-directional Link Bandwidth (BLAB) is defined as the minimum of the aggregate memory bandwidth, aggregate bus bandwidth, or the sum bi-directional link peak user payload data bandwidth.

## 2.2 OpenIB Access Layer [MR]

The Offeror's IB software stack shall include software developed to the OpenIB Access Layer. This work shall be part of the OpenIB (www.openib.org) community effort.

### 2.2.1 Low Latency Interrupt Mode [TR-1]

The Offeror's delivered OpenIB software stack kernel and user space latency for interrupt (non-polling) style communications in the OpenIB access layer should be within 10% of that for polling mode.

### 2.2.2 Kernel and User Space Verbs API Interfaces [TR-1]

The OpenIB Access Layer portion of the IB software stack should include both the Verbs API kernel and user space interfaces (data path) whose functionality is defined in the InfiniBand architecture specification volume 1 chapter 11 (www.infinibandta.org). The Access Layer should also include a Verbs framework with clear boundaries between standards-compliant Verbs and vendor extensions.

### 2.2.3 Integrate open source Verbs API patches for stability and performance enhancements into main OpenIB Verbs API stack [TR-1]

The Offeror should integrate any Verbs API patches into the OpenIB software stack. These Verbs API patches should include future work as well as recently open sourced patches that have not been integrated into the OpenIB software stack.

### 2.2.4 High Performance RC Send/Recv [TR-1]

The Offeror should deliver OpenIB reliable connections (RC) send/receive operations that will obtain 90% of the LAB of the supported IB devices. Both the kernel and user space interfaces should deliver this level of performance.

### 2.2.5 High Performance RC RDMA read/write Operations [TR-1]

The Offeror should deliver OpenIB reliable connection (RC) RDMA operations that will obtain 95% of the LAB of supported IB devices. Both the kernel and user space interfaces should deliver this level of performance.

### 2.2.6 High Performance RC Atomic Operations [TR-1]

The Offeror should deliver OpenIB reliable connection (RC) atomic operations (fetch and add, compare and swap) that should obtain 3 µs (micro-second) latency on supported IB devices. Both the kernel and user space interfaces should deliver this level of performance. The latency should be measured from a user process on one node to another user process on any other node in the cluster.

### 2.2.7 High Performance UD Send/Recv [TR-1]

The Offeror should deliver OpenIB unreliable datagram (UD) send/receive operations that will obtain 90% of the LAB of supported IB devices. Both the kernel and user space interfaces should deliver this level of performance.

### 2.2.8 Design OpenIB access layer to allow ease of portability to multi-vendor hardware [TR-1]

The Offeror's OpenIB access layer should be designed to be portable to IB host channel adapters, switches, gateway, and router devices from multiple vendors. The Offeror should describe in the SOW response the proposed strategy for designing the OpenIB access layer in order to facilitate portability.

### 2.2.9 Supported Host IO Interfaces [TR-1]

The Offeror's delivered OpenIB access layer should support host channel adapters with PCI Express, HyperTransport and PCI-X 64b/133Mhz interfaces. The Offeror should describe the proposed strategy for designing the OpenIB access layer in order to obtain the performance levels described in this SOW from these host IO interfaces.

### 2.2.10 Multiple HCA per SMP [TR-1]

The Offeror's OpenIB access layer software should support a scalable design for multiple host channel adapters per SMP node. The Offeror should describe the proposed strategy for designing the OpenIB access layer in order to obtain near-linear scaling from multiple HCAs per SMP node.

### 2.2.11 Reliable Multicast [TR-1]

The Offeror's OpenIB access layer software should support scalable reliable multicast. The Offeror should describe the proposed strategy for designing and implementing scalable and high performance reliable multicast.

### 2.2.12 OpenIB Access Layer Latency [TR-1]

The Offeror's OpenIB access layer interface software overhead should be less than 1 µs as measured by sending a minimum length message from a user process on one node to another user process on any other node in the cluster and receiving back an acknowledgment divided by two (ping-pong latency).

### 2.2.13 Extensible API and API implementation to SM, SA, and CM services for user and kernel space applications and modules [TR-1]

The Offeror's OpenIB access layer should provide programmer friendly API's to Subnet Management, Subnet Administration, and connection Management services. These API's should be implemented and provided for both user and kernel space applications. The API's should be easily extensible where service extensions to the InfiniBand standard are clearly denoted by their function name.

### 2.2.14 Enhancements to Verbs API for URDMA [TR-2]

The Offeror's OpenIB software stack should include enhancements to the software transport verbs to define and implement an unacknowledged, Unreliable RDMA capability. This capability is denoted URDMA. The Offeror should work with the IBTA to get these or similar URDMA enhancements accepted into the InfiniBand architecture specification. If accepted by the IBTA, consistent with this project's duration, the Offeror should modify the implemented URDMA capability in the delivered software stack.

### 2.2.15 Enhancements to Verbs API for UD Atomic Operations [TR-2]

The Offeror's OpenIB software stack should include enhancements to the software transport verbs to define and implement an unacknowledged, Unreliable atomic capability. This capability is denoted UD atomic. The Offeror should work with the IBTA to get these or similar UD atomic enhancements accepted into the InfiniBand architecture specification. If accepted by the IBTA, consistent with this project's duration, the Offeror should modify the implemented UD atomic capability in the delivered software stack.

### 2.2.16 High Performance URDMA Operations [TR-2]

The Offeror should deliver OpenIB unreliable datagram (UD) RDMA operations that will obtain 95% of the LAB for the supported IB devices. Both the kernel and user space interfaces should deliver this level of performance.

### 2.2.17 High Performance UD Atomic Operations [TR-2]

The Offeror should deliver OpenIB unreliable datagram (UD) atomic operations (fetch and add, compare and swap) that should obtain three µs (micro-second) latency on supported IB devices. Both the kernel and user space interfaces should deliver this level of performance. The latency should be measured from a user process on one node to another user process on any other node in the cluster.

### 2.2.18 High Performance Gather/Scatter operations [TR-2]

The Offeror's delivered OpenIB access layer should obtain 95% of the BLAB for scatter/gather transfers with large memory segments. Transfers with small segments should be no slower than individually posted work requests.

### 2.2.19 Support for service level to virtual lane mapping, partitioning, automatic path migration, and multi-path routing [TR-2]

The Offeror should describe the proposed strategy for designing and implementing an OpenIB access layer that will expose service level to virtual lane mapping, partitioning, automatic path migration, and multi-path routing features.

## 2.3 InfiniBand Network Management [MR]

The Offeror's OpenIB stack shall include components for Subnet Management (SM), Subnet Administration (SA), Connection Management (CM), and HPC routing. High performance computing imposes unique requirements for performance and scalability on the SM, SA, CM, and routing. The Offeror's OpenIB stack shall include a scalable SM and optimized algorithms for fabric initialization and connection establishment. Routing algorithms optimized for high performance computing are also critical for enabling application scalability to thousands of nodes. The SM and CM scalability shall require maintaining fast fabric initialization and connection establishment across fat-tree and Clos networks with at least 4,096 host ports. The Offeror shall make all additions and modifications to the existing OpenSM found at www.openib.org.

### 2.3.1 Subnet Manager and Subnet Administrator for HPC Clusters [TR-1]

The Offerors delivered IB Subnet Manager (OpenSM) and Subnet Administration (SA) components should be fast, scalable and reliable. The Offeror's OpenSM should initialize a single switch configuration of 144 port or greater within 2 seconds. The OpenSM is scalable if it can initialize a fat-tree or Clos network configuration with 4,096 host ports in less than 30 seconds with no knowledge of the IB fabric, and in less than 15 seconds given the topology information. The OpenSM should utilize speculative assignment of historical LIDs to avoid rediscovery of full topology at every reboot and the fast dump and restore of routing tables. To meet the scalability

requirements for OpenSM, the Offeror's OpenSM should be a distributed subnet manager rather than a centralized subnet manager. The SA is scalable if a single SA instance supports an aggregate performance of at least 200 queries per second for 4096 nodes. All administration functions should be accessible from a Linux command line and scriptable utilizing EXPECT (http://expect.nist.gov/) and regular expressions to denote groups of components. Both the OpenSM and SA components are reliable if they successfully complete their respective operations or return correct error codes except for one in ten thousand (1 in $1x10^4$) attempts. Both the OpenSM and SA interfaces should allow system administrators to issue multiple commands in a single transaction. Each command should be executed within 30 seconds.

### 2.3.2 Fast and Scalable Connection Manager [TR-1]

The Offeror's delivered Connection Manager (CM) should configure an all-to-all connection topology in a single 288 port configuration with 1152 processes (4 processes per node, and 1 port per node) in less than 5 seconds without connection caching, and within 2 seconds with connection caching. The Offeror's delivered CM should configure an all-to-all connection topology in a 4,096 port fat-tree configuration with 16,384 processes (with 4 processes per node, and 1 port per node) in 60 seconds or less without connection caching, and in 15 seconds or less with connection caching.

### 2.3.3 Network topology awareness and verification services [TR-2]

The Offeror's OpenSM and SA should provide services for topology information and verification services to diagnostic and management tools, and upper layer protocols such as MPI. The information should be provided in the form of a text based list of the following: number of host HCA's, port number on switch, internal ASIC on the switch that each port on the HCA is a attached to, blade number on switch, switch number, neighboring nodes on this ASIC, neighboring node on this blade, any facilities information that can be input (rack number, placement, cable labels, etc.), link speed of each of the HCA connections, number of active links per HCA, number of active links per ASIC, link speed of active links per ASIC, number of active links per switch blade, number of active links per switch, and number of network planes.

The information should be defined in the section on Scalable InfiniBand Diagnostics and Management tools and should be available in the various forms necessary for any complete diagnostic software. It is also necessary for node allocation from MPI and location awareness from within MPI. Issues of placement and awareness can be used to improve load balancing as well as assessing potential performance issues. This in the general case answers the question needed by many applications of neighborhood definition as well as self discovery. Allowing MPI the necessary information to properly place jobs on the correct nodes with respect to their co-workers should lead to performance increases. The Offeror's OpenSM and SA should provide this information and access to it from a Command Line Interface (CLI), an Application Program Interface (API), and a Graphical User Interface (GUI). These tools should be used for High Level Languages as well as for the basis of more complete and robust diagnostics.

From the information supplied in any of the necessary forms, the Offeror should provide derived information that can be aggregated into options like Distance Graphs. A Distance Graph is defined as follows: Let D be a set of positive numbers containing 1, then the D-distance graph X (D) on a nonempty subset X of Euclidean space is the graph with vertex set X and edge set, where d(x,y) is the Euclidean distance between vertices x and y. Various visual displays of the topology of the interconnect fabric should be generated when the information is provided. This should be generated and verified using various Open Source graph packages, like Boost Graph Library which can be found at www.boost.org. Graphical displays of the system fabric with "drill down" capabilities, color enhanced coding for failed modules and constant link state and speed monitoring are considered very desirable. Additional information that can be generated should be graphs and

information concerning routing analysis, route cost analysis, best route placement, optimized access patterns based on application requirements as well as many others.

Another very important requirement is plug & play for rapid updates and verification to all of the information when any element of the environment is perturbed. Verification of uniqueness of Node ID's, GUID's, etc. must be maintained in the event of incomplete system verification of replacement parts or modules.

Verification services may use the diagnostics delivered in section 2.3

### 2.3.4 Optimized routing algorithms for High Performance Computing [TR-2]

The Offeror's delivered subnet manager and administrator should have optimized routing algorithms for fat-tree and Clos topologies. The routing algorithms should reduce network congestion and hot-spots in standard static routing. Suggestions for dynamic and adaptive routing extensions to the InfiniBand standard should also be considered.

## 2.4 Scalable InfiniBand Diagnostic and Management Tools [MR]

Scalable InfiniBand diagnostic and management tools are essential to providing a stable cluster environment and to minimizing system testing and integration time. Diagnostic tools shall be able to accurately diagnose failures of the InfiniBand network and hosts, to verify cluster topology, and to explore InfiniBand fabric setting and routing. Management tools shall allow network booting over InfiniBand, and provide tools for distributed firmware installations across cluster switch and nodes.

### 2.4.1 Fast, Scalable and Reliable Interconnect Diagnostics [TR-1]

The Offeror's delivered OpenIB software stack should have scalable and reliable diagnostics (command line utilities and libraries) that should determine defective cables, cable connectors, ASIC ports, switch ports, and host channel adapters. These diagnostics should have the capability to determine component failure down to Field Replaceable Units (FRU). These diagnostics are fast if they can reliably diagnose problems for single switch configurations within 10 minutes. These diagnostics are scalable if they can reliably diagnose problems in fat-tree and Clos switch configurations with 4,096 node ports in less than 60 minutes. These diagnostics are reliable if they complete the diagnosis process with false-positive or false-negative errors reported that occur less than one in one thousand (1 in $1x10^3$) attempts. The diagnostic tools should have the ability to check for HW and SW versions consistency and compatibility.

Diagnostic tools should use the OpenSM and SA defined in section 2.3.1, the CM defined in section 2.3.2, and the network topology and verification services defined in section 2.3.3

The diagnostic tools (application codes and libraries) should have error messages that may be easily interpreted by University personnel with minimal IB training.

Hardware diagnostic tools should consist of an Open Source API and Open Source command line tools utilizing published interfaces to vendor specific (potentially proprietary) hardware features. The published interfaces to vendor specific (potentially proprietary) hardware features should be provided in binary only form.

### 2.4.2 Host Side Diagnostic and Management Tools [TR-1]

The Offeror's delivered OpenIB software stack should have host side diagnostic tools. These diagnostic tools should be capable of obtaining information on how many HCAs are in the node, how the node is configured, the state of the IB drivers, the number of network planes the node is attached to, the speed of IB links, implemented services, route configurations, and performance. The host management tools should support scalable network flash of HCA firmware. All diagnostic and management information should be accessible through an API and CLI. The tools should be

capable of obtaining network performance counters and statistics at the node level. The tools should be capable of probing the device driver state, and of performing HCA and software version consistency checks. The host side diagnostics should use the OpenSM, SA, CM, and tools defined in sections 2.3.1-2.3.3.

### 2.4.3 Fabric Explorer and Debugging Tools [TR-1]

The Offeror's OpenIB stack should include the tools vping, openibdump, openibroute, and openibwalk. Vping may be for non-ip based communication protocols. Openibdump should be a tcpdump for IB. Openibroute should test each piece of silicon along a route and report back status of the components and latencies along paths. Openibroute should use all routes to all potential destinations. Openibwalk should use a specified route table to verify that routes exist and that all destinations are reachable. The Offeror's OpenIB stack should also support the standard ip tools (for example, tcpdump, traceroute, ping, and netperf) running over IPoIB and SDP.

### 2.4.4 Fabric performance test suite for scaling, bandwidth, and latency [TR-2]

The Offerors delivered OpenIB software stack should include software to test the performance, scaling, and congestion management attributes of the IB infrastructure.

### 2.4.5 InfiniBand Route Visualization [TR-2]

The Offerors delivered OpenIB software stack should include software to display IB route information in a visual presentation format (pictures, not words or numbers) that is easily understandable by someone who is not an expert in IB technology.

### 2.4.6 Cluster Topology Visualization Tools [TR-2]

The Offerors delivered OpenIB software stack should include software to discover and display IB component and topology information in a visual presentation format (pictures, not words or numbers) that is easily understandable by someone who is not an expert in IB technology.

### 2.4.7 Topology verification tools [TR-2]

The Offerors delivered OpenIB software stack should include software to discover and verify IB component and topology information.

## 2.5 InfiniBand Upper Layer Protocols [MR]

The Offeror's OpenIB software stack shall include the following HPC upper layer protocols: IPoIB, SDP, MPI-2, and Sandia Portals. The intent of these requirements is to add InfiniBand support for the existing Open Source MPI-2 implementations, and not to develop a new MPI-2 library. The intent of these requirements for Sandia Portals is limited to integrating/porting existing InfiniBand Portals implementations into the OpenIB software stack (www.openib.org).

### 2.5.1 High Performance IPoIB [TR-1]

The Offeror's OpenIB software stack should support IP over InfiniBand (IPoIB) protocol as defined in InfiniBand specification available at www.infinibandta.org. The delivered IPoIB implementation may achieve 75% of aggregate link bandwidth with less than 25% of a single CPU utilization.

### 2.5.2 IPoIB with DHCP [TR-2]

The Offeror's IPoIB should support DHCPv6 as defined in (http://www.ietf.org/rfc/rfc3315.txt).

### 2.5.3 High Performance SDP [TR-1]

The Offeror's OpenIB software stack should include Socket Direct Protocol (SDP) protocols as defined in the InfiniBand specification available at www.infinibandta.org. The delivered SDP

implementation should achieve 90% of aggregate link bandwidth with less than 25% of a single CPU utilization.

### 2.5.4 SDP transparent sockets support [TR-2]

The Offeror's SDP should use an optional socket offload modules for transparent support for user and kernel sockets.

### 2.5.5 Sandia Portals [TR-1]

The Offeror's OpenIB software stack should include the Sandia Portals protocol (http://sourceforge.net/projects/sandiaportals). The intent is to integrate/port an existing Sandia Portals InfiniBand implementation into the OpenIB software stack. The Sandia Portals InfiniBand implementation should support RDMA operations.

### 2.5.6 Open Source MPI 2.0 [MR]

The Offeror's OpenIB software stack shall include the Message Passing Interface 2.0 (MPI-2) defined in (http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html) except for dynamic tasking. Performance enhancements shall be implemented in a general fashion to benefit any communicators, data type objects, and blocking and non-blocking point to point communications. It is anticipated that the MPI-2 one-sided communications shall be provided, but not optimized.

#### 2.5.6.1 NAL for MPI-2 [TR-1]

The Offeror's OpenIB MPI-2 library should have a network abstraction layer (NAL). This layer should abstract all interactions between the upper layers of MPI library and the OpenIB software stack. This NAL should provide a framework for implementing separate network modules, for example InfiniBand, Ethernet, Quadrics ELAN4 or Myrinet, with this SOW only focusing on the InfiniBand module.

#### 2.5.6.2 MPI-2 Infrastructure [TR-1]

The Offeror's OpenIB MPI-2 library should use the Offerors OpenIB CM, SA, OpenSM component as appropriate. In particular, the MPI-2 library should use the OpenIB CM to manage connections.

#### 2.5.6.3 Low Latency MPI-2 Interrupt Mode [TR-1]

Latency for interrupt (non-polling) style communications in MPI-2 library should be within 10% of that for polling mode.

#### 2.5.6.4 High Bandwidth MPI-2 [TR-1]

The sustained single task and aggregate MPI message bandwidth delivered to/from each node should be at least 90% of the LAB (i.e., 90% when sending/receiving messages of a size that optimizes system performance with MPI_SEND, MPI_RECV or MPI_ISEND, MPI_IRECV pairs) and 90% of the BLAB (i.e., 90% with MPI_SENDRECV or MPI_ISENDRECV pairs).

#### 2.5.6.5 Low Latency MPI-2 [TR-1]

The MPI-2 library latency, as measured by sending a minimum length MPI message from a single MPI task user program memory on one cluster node to one other MPI task user program memory on any other node in the cluster and receiving back an acknowledgment divided by two (standard MPI user space MPI_Send/MPI_Recv and MPI_Isend/MPI_Irecv ping-pong test), should be less than 3.0 micro-seconds ($3x10^{-6}$ seconds) for PCI-Express adapters and 5.0 micro-seconds ($5x10^{-6}$ seconds) for PCI-X 64b/133 MHz adapters.

The MPI-2 library latency, as measured by sending a minimum length MPI message from a single MPI task user program memory on one cluster node to one other MPI task user program memory on the same node and receiving back an acknowledgment divided by two (standard MPI user space MPI_Send/MPI_Recv and MPI_Isend/MPI_Irecv ping-pong test), should be less than 1.0 micro-seconds ($1x10^{-6}$ seconds).

When measured between any two nodes in the system with one MPI task per CPU on each node for 8-way or smaller SMPs, the MPI user space ping-pong latency should be less than 5.0 micro-seconds ($5x10^{-6}$ seconds) for PCI-Express adapters and should be less than 10.0 micro-seconds ($1x10^{-5}$ seconds) for PCI-X 64b/133 MHz adapters.

#### 2.5.6.6 High Performance and Scalable MPI Collective Operations [TR-1]

The delivered MPI-2 library should support scalable global operations under normal operating conditions. This should be measured by the GLOBAL_OP benchmark. That is, the GLOBAL_OP benchmark should measure at least the following MPI global operations: MPI_Allreduce, MPI_Reduce, MPI_Broadcast, MPI_Barrier, and MPI_Alltoall. The Offeror's MPI-2 library should implement all the listed global operations utilizing InfiniBand hardware based multicast and atomic primitives. The MPI global operations should be measured with the following methodology: for a given task count, iterate 10,000 times over doing one second of floating point work followed by the MPI global operation with one 64b floating point value and repeat. The runtime per iteration should be measured as a function of MPI task count (NTASK) from 2 up to the number of maximum number of MPI tasks supported on the system. The runtime per iteration as a function of NTASK should remain constant or increase by at most the log2(NTASK). This benchmark should be run under conditions matching those of the general workload (i.e., special calls requiring root access that perform task binding to CPUs or changing thread/process/task priorities within the application is specifically not allowed) with normal system daemons running under normal operating conditions.

#### 2.5.6.7 Optimize MPI data-types communication to achieve high performance for non-contiguous data transfers [TR-2]

Delivered bandwidth for MPI-2 for non-contiguous data transfers should be no more than 15% slower than for contiguous operations with the same number of bytes in the payload.

#### 2.5.6.8 Performance Scalability[TR-2]

The MPI implementation should permit scalability to cluster sizes anticipated by NNSA. Furthermore, overhead associated with the MPI layer should remain a set percentage of the underlying fabric capability regardless of communicator size. Care should be taken to avoid built-in limitations to scalability such as fixed size tables and serial bottlenecks. Dynamic allocation of resources should be used to allow MPI to adapt naturally to the resource requirement. The validation test for this requirement should be an all-to-one of 4,096 MPI tasks writing 1 MB buffers. For 4,096 MPI tasks the performance metric should be 90% LAB for the receiving node.

#### 2.5.6.9 Memory Scalability [TR-2]

The delivered MPI-2 library buffers should be designed so that the buffers requirement is not dependent on the maximum size of messages and should be a fixed amount (buffer pool approach) or an amount that grows slower than linear as a function of the number of MPI tasks in the job. The delivered MPI-2 library buffers should be managed so that an application can control a portion of buffer space required for point-to-point and all-to-all communications. In

particular, the University desires that if an application guarantees that receives are posted before sends on point-to-point messages, then it be possible to avoid MPI staging buffers completely. The Offeror should deliver (electronic) written documentation that describes the performance features of the MPI implementation for each software release on the delivered hardware. All environmental settings that impact MPI operation, buffering and performance and their impact to 64b user applications performance should be tested and their effectiveness and reliability documented.

#### 2.5.6.10 MPI-2 Thread Safety [TR-2]

The MPI-2 library should be a thread compliant implementation allowing multiple threads within an MPI task to make MPI communication calls. The delivered MPI should support programming models used by ASC code teams such as MPI everywhere and a mix of MPI and threads (explicit thread calls or OpenMP). The Offeror should provide MPI-2 design documentation describing the thread safety implementation.

#### 2.5.6.11 Multiple HCA per SMP [TR-2]

The Offeror's MPI-2 library should support a scalable design for multiple host channel adapters per SMP node. The Offeror's MPI-2 library should support single and multiple network planes. (The Offeror should describe in their proposal, the proposed strategy for designing the MPI-2 library in order to obtain near linear speed up from multiple HCAs per SMP node.)

#### 2.5.6.12 Fast, Reliable, Scalable MPI Job Launch [TR-3]

The Offeror's MPI-2 implementation should include a mechanism for fast, reliable, scalable MPI job launch. The job launch capability should be a Linux command line interface that allows non-root users to launch jobs. The University will approve the Offerors MPI-2 job command line user interface design. The Offeror should describe the architecture of the MPI-2 job launch facility in their proposal. The job launch facility is fast if it launches a 4 task MPI job on 4 nodes in less than 10 seconds. The job launch facility is scalable in time if large job launch is sub-linear in the number of tasks launched and is on average less than 300 seconds with standard deviation less than 10 seconds for 4,096 MPI task jobs when measuring 200 successive jobs. This time should be measured with a job that calls MPI_init(), MPI_Finalize(), and then terminates. The job launch facility is reliable if it successfully completes or returns correct error codes correctly except for one in one billion (1 in 1x109) attempts. The MPI job launch facility should utilize in-band IB communications mechanisms.

#### 2.5.6.13 High Performance and Scalable MPI Gather/Scatter [TR-3]

The delivered MPI-2 library gather/scatter operations should obtain 95% of the BLAB for scatter/gather transfers with large memory segments. Transfers with small segments should be no slower than individually posted work requests.

The performance for a single segmented MPI-IO file should be within 90% of the bandwidth of a similar segmented file generated by the POSIX interface for the University provided test program IOR. The intent of this requirement is to ensure efficient generation and communication of distributed data types such as those commonly found with MPI-IO programs.

## 2.6 High Performance Computing Stability and Robustness

HPC environments require stable and robust software environments.

### 2.6.1 Stability [TR-1]

The Offeror's delivered OpenIB Software Stack should be able to sustain a workload consisting of 50 or more parallel applications with task sizes ranging from 16 to 4,096 and with run times ranging from 1 minute to 24 hours for 7 days without interruption of applications and without loss or corruption of user or system data due to Offerors provide OpenIB software stack.

### 2.6.2 Robustness [TR-1]

The Offeror's OpenIB Software Stack should be robust. The software stack is robust if the variations in application runtime are less than 5% of their overall runtime and the software can handle a wide range of application IB usage patterns and the software can handle hardware Field Replaceable Unit (FRU) by gracefully degrading the network and not causing catastrophic network unavailability under FRU failure. Other attributes of robust software stack include being able to mix components (switches and adapters) from multiple IB vendors in a single fat-tree or Clos switching network for large clusters.

# 3 SOFTWARE DEVELOPMENT

The following requirements deal with the types of systems that need to be supported and the software development and testing practices for the project.

## 3.1 Platform and OS Independence and Portability [MR]

Platform independence and portability is critical to maintaining a flexible HPC software environment. Support for the following environments shall be requirements (if designated with a MR) of the open source software regression testing suite. The InfiniBand software support matrix for various versions of the Linux kernels (2.6 and RHEL), gcc, glibc, POSIX thread libraries, and other core system components shall be revised on an ongoing basis by the University and the successful Offeror. The support matrix shall include (if designated with a MR), but is not limited to, the following areas.

### 3.1.1 Tracking Latest 2.6 Kernels [MR]

The Offeror's OpenIB Software Stack development shall track and support the major 2.6.x kernel releases at kernel.org. The stack shall be integrated into the Linux 2.6 kernels at kernel.org. The Offeror's OpenIB Software Stack production release candidates shall support Linux clusters using one 2.6.x kernel and operating system image per node (with or without disks).

### 3.1.2 Tracking Latest RHEL Kernels [TR-1]

The Offeror's OpenIB Software Stack development should track and support the major RedHat Enterprise Linux (RHEL) releases. The stack should be integrated into the RHEL kernels. Offerors OpenIB Software Stack production release candidates may support Linux clusters using one RHEL kernel and operating system image per node (with or without disks).

### 3.1.3 Tracking BProc Linux [TR-1]

The Offeror's OpenIB Software Stack and kernel work should support Linux clusters using the BProc single system image software found at bproc.sourceforge.net and www.clustermatic.org. The Offeror's OpenIB Software Stack should support BProc 4.0, and continue to track future BProc releases within the period of performance of the proposed Subcontract.

### 3.1.4 Linux Normal Format Coding Style [TR-1]

The Offeror should follow the Linux kernel coding style. The coding style guidelines can be found in the Documentation/CodingStyle file in the Linux 2.6 kernel code tree.

### 3.1.5 Support for Intel IA32 architecture [TR-1]

The Offeror's OpenIB Software Stack should be deployed and tested on the Intel IA32 architecture.

### 3.1.6 Support for Intel and AMD x86-64 [TR-1]

The Offeror's OpenIB Software Stack should be deployed and tested, with 32 and 64 bit applications on a 64 bit operation system, on the AMD Opteron and the Intel EM64T (Xeon-64) architectures.

### 3.1.7 Support for Intel Itanium (IA64) [TR-1]

The Offeror's OpenIB Software Stack should be deployed and tested, in 64-bit mode, on the Intel Itanium (IA64) architecture.

### 3.1.8 Tracking Latest SuSE Kernels [TR-3]

The Offeror's OpenIB Software Stack development should track and support the major SuSE Linux releases. The stack should be integrated into the SuSE kernels. Offerors OpenIB Software Stack

production release candidates should support Linux clusters using one SuSE kernel and operating system image per node (with or without disks).

#### 3.1.9 Support for IBM PowerPC 970 [TR-3]

The Offeror's OpenIB Software Stack should be deployed and tested, for 32 and 64-bit applications on a 64-bit operating system, on the IBM PowerPC 970 architecture.

### 3.2 Open Source Software [MR]

All software developed under this Subcontract shall be released under a University approved Open Source license (see the Sample Subcontract) and delivered to the University with source code.

#### 3.2.1 Documentation [MR]

End user documentation, component interface documentation, application programming interface (API) documentation and software test suites should be provided as part of the open source release. The methods used for documentation and testing should be broadly available—not requiring proprietary products with limited access. There shall be documented tunables in all provided software. List tunables, definitions and locations, recommended settings and how to change tunables, impact of changes of tunables, dependencies among tunables.

#### 3.2.2 Source Code Documentation System [TR-1]

The Offeror's OpenIB Software Stack should use a source code documentation system such as doxygen (www.doxygen.org). Running 'make documentation' at any level should process all doxygen tags produce documentation for that code.

#### 3.2.3 Access to Source Code Repository [TR-1]

Read/Write access to the software source repository should be provided to individuals, designated by the University, at LLNL, LANL, and SNL. Read access to the software should be broadly available. Source code should be provided in a form that can be installed at openib.org under the Subversion revision control system.

### 3.3 Software Build Environment [TR-1]

The following target requirements deal with the tools and environment the Offerors Software Stack may be developed with.

#### 3.3.1 Development and Testing with GNU compilers [TR-1]

The Offerors delivered OpenIB Software Stack should compile, load and run utilizing the GNU compilers and standard Linux code development tool chain (compilers, loaders, etc). The Offerors delivered OpenIB Software Stack, testing harness and application tests should be compiled and run utilizing the GNU compilers and standard Linux code development tool chain and utilities (e.g., bash for scripts) for all testing done for this SOW.

#### 3.3.2 Support for NPTL [TR-1]

The Offerors delivered OpenIB Software Stack should support the Native POSIX thread library (NPTL). Documentation can be found at http://people.redhat.com/drepper/nptl-design.pdf

#### 3.3.3 Build and Install Requirements [TR-1]

Offeror delivered software should be manipulated (build, install, and other operations) with Linux “make” utility utilizing Linux style “makefiles”.

Software may be installed by running “make install”. Tags should be built in a given directory and subdirectories by running “make tags and/or “make etags”. Shell scripts are not an acceptable alternative to makefiles.

For hierarchical build trees it should always be possible to build a subtree by running make in the head of that subdirectory.

#### 3.3.3.1 Target Directory Independence [TR-1]

Offerors Software Stack, tools, and diagnostics should be constructed so that the end-user may set the head directory for source code, binaries, libraries, kernel modules, documentation and other components. In other words, Offerors Software Stack should be constructed so it may not require special path names or other built-in assumptions.

#### 3.3.3.2 Linux Kernel Tree Independence [TR-1]

The Offeror’s InfiniBand kernel drivers build and install mechanisms should be constructed so that the drivers are buildable both inside and outside the Linux kernel tree structure.

#### 3.3.3.3 GNU Autotools [TR-1]

The Offerors OpenIB software stack should utilize GNU Autoconf, Automake, and Libtool to enable multiple platform support. The GNU Autotools should provide a superset of capabilities needed to support all other target requirements for supporting build environments for several different Linux distributions. See http://sources.redhat.com/autobook/ for more information.

#### 3.3.3.4 LSB Compliance [TR-2]

The Offerors OpenIB stack should be compliant with the Linux Standard Base (LSB) that specifies an interface between an application and a run-time environment. Many distributions have achieved LSB certification for their run-time environments. The LSB specification should be obtained at http://www.linuxbase.org/spec/.

## 3.4 Extensibility and Usability of Design [MR]

Documentation specifications of system interfaces and extensibility interfaces shall be provided to enable extending the work to other systems.

### 3.4.1 Ease of Extensibility [TR-1]

New features should be easy to add, both by the Offeror and by other collaborators working with the software. (The Offeror’s proposal should describe the software architecture that makes this possible. Examples are extensions to Verbs and other access layer components, to management tools and diagnostics, and to MPI-2, SDP, IPoIB, and Sandia Portals.

### 3.4.2 Interface Usability Testing [TR-1]

The Offeror should provide an interface usability testing plan that involves tests with University designated users on real applications before the interfaces are finalized.

## 3.5 Software Maintenance and Support [MR]

The Offeror shall provide maintenance and support of the OpenIB Software Stack for the duration of period of performance of the proposed Subcontract.

The Offeror shall designate a point of contact that shall have full knowledge of the Offerors Open Source software stack and shall be able to diagnose problems in the field and perform modifications as

appropriate and required. The Offerors point of contact shall be approved by the University Technical Representative.

### 3.5.1 Software Support Mechanisms [TR-1]

Software bugs should be tracked by the GNATs or Bugzilla tracking mechanism. Software bugs should be prioritized as: Severity 1, Severity 2, Severity 3, or Severity 4. Software bugs should only be prioritized by the University Technical Representative (UTR) or other designated University staff. Software bugs should only be declared as fixed or resolved by the UTR or other designated University staff. The various Severity levels should be defined as follows:

- A Severity 1 problem is a catastrophic problem that may severely impact the University's ability to conduct business. This may mean that the University's systems and/or InfiniBand hardware or software are down or not functioning and no procedural workaround exists.
- A Severity 2 problem is a high-impact problem in which the University's operation is disrupted but there is capacity to remain productive and maintain necessary business-level operations. The problem may require a fix be installed on the University's system prior to the next planned release of InfiniBand hardware or software.
- A Severity 3 problem is a medium-to-low impact problem that involves partial, but non-critical functionality loss. This category of problem impairs certain operations but generally allows the University to continue to function. This may be a minor issue with limited loss or no loss of functionality or impact to the University's operation.
- Severity 4 shall be assigned to general usage questions, recommendations for future product enhancements or modifications or to calls that are provided to InfiniBand for information purposes. There is no impact on the quality, performance or functionality of the software.

### 3.5.2 Software Support Response [TR-1]

Severity 1 problems should be addressed not later than one working day after submission. The Offeror should begin diagnosis of the problem with the University during the one day period. The Offeror should commit all necessary resources to work on Severity 1 problems until such problems are fixed. The Offeror should interact with University staff on an ongoing basis when any outstanding Severity 1 problems exist. Severity 2 problems should be addressed not later than two working days after submission. The Offeror should begin diagnosis of the problem with the University during the two day period. The Offeror should commit all necessary resources to resolve all Severity 2 problems within one week. The Offeror should interact with University staff on a weekly basis when any outstanding Severity 2 problems exist. Severity 3 and Severity 4 bugs should be addressed on an ongoing basis with weekly status reports.

## 3.6 Testing Environment [MR]

OpenIB stack release candidates with HPC capabilities shall be evaluated on DOE InfiniBand clusters at Los Alamos, Sandia, and Lawrence Livermore National Laboratories. The following list of components is available for testing within the Tri-Laboratory community. Only components available at this time and based on publicly announced products are listed. Other environments are planned or are currently available based on unannounced products. The Offeror shall plan to test utilizing these environments. In addition, Offeror may propose other environments for testing of the OpenIB software stack. The

proposals shall describe in detail the support matrix proposed for testing for at least the first deliverables (section 4.2).

**IBA RFP SOW**
**LIST OF SUPPORTED SW/HW**

Version 2
May 28, 04

| | | | | | | SuSE SLES | RHELV3 | 2.6.x | X86-64 | IA-64 | IA-32 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Chip Sets** | | | | | | | | | | | |
| TR-1 | ServerWorks | Grand Champion LE, HE | PCIX 64b/133MHz | IA-32 | ADEV, RoSE | X | | | | | X |
| TR-1 | | Grand Champion SLX | PCIE x8 | x86-64 | | | | | | | |
| TR-1 | Intel | E750x | PCIX 64b/133MHz | IA-32 | MDEV, BIGDEV, CAT | X | | X | | | X |
| TR-1 | | i8870 | PCIX 64b/133MHz | IA-64 | TDEV | X | | | | X | |
| TR-1 | | E2772 (Tumwater) | PCIE x16, x8 | x86-64 | GALOIS | | | X | | | |
| TR-1 | | E7520 (Lindenhurst) | PCIE x8 | x86-64 | | | | | X | | |
| TR-1 | AMD | 8111/8131 | PCIX 64b/133MHz | x86-64 | BLUESTEEL, PANTA | X | | X | X | | |
| TR-2 | nVidia | CK8-04/IO4 | PCIE x8 | x86-64 | | | | | X | | |
| **HCA Interfaces** | | | | | | | | | | | |
| TR-1 | IBA x4 | Single & Dual | PCIX 64b/133MHz | | BLUESTEEL, CATALY | X | | X | X | X | X |
| TR-1 | IBA x4 | Single & Dual | PCIE x8 | | GALOIS | | | X | X | | |
| TR-2 | IBA x12 | Single | PCIE x16 | | | | | | | | |
| TR-3 | IBA x12 | Single & Dual | PCIE x16 | | | | | | | | |
| **IBA Switches** | | | | | | | | | | | |
| TR-1 | Mellanox | 24 (IS3), 144 (IS3) port IBA x4 | | | CADILLAC, CATALYST | | | X | | | X |
| TR-2 | Voltaire | ISR 9024, 9600, 9288 IBA x4 | | | CADILLAC, CATALYST, RoSE | | | X | | X | X |
| | Mellanox | 96 port (IS2) IBA x4 | | | BLUESTEEL | | | X | | | X |

# 4 PROJECT MANAGEMENT

Execution of this project requires close coordination between the University and the Offeror and the OpenIB community. The following project management requirements and milestones are intended to facilitate the successful completion of the project within budget and the specified schedule.

## 4.1 Project Plan [MR]

The Offeror shall deliver an overall project plan describing the development process that shall be utilized in the project. The project plan shall also describe the oversight role of the University and how the project shall be coordinated with the OpenIB effort. The project plan shall include weekly technical telecons with the project participants, monthly status meetings or telecons to review technical progress on specific detailed topics and quarterly face-to-face meetings for overall management reviews. The project plan shall consist of two parts: 1) a Microsoft Project GANNT chart describing at least three levels of details and assigning resources and estimated durations and dependencies between parts of the project; 2) a written document describing the management of and strategy for executing the project. The written project plan document shall indicate the project members and management structure. The project plan shall include both development and testing activities.

## 4.2 Milestones [MR]

All milestones described below are suggested in order to facilitate the Offerors planning of the project. Offeror may substitute a different milestone schedule, but a milestone schedule (with associated dates and payments) must be included in the Offerors proposal. All dates referenced in the following milestones are durations after the date the contract is executed.

### 4.2.1 Project Plan (1 Month)

The Offeror shall deliver a project plan meeting the requirements in section 4.1. In addition, the Offeror shall complete the formation of the project and allocate resources according to the project plan. This milestone is complete when the University Technical Representative (UTR) verifies that the project has been formed, personnel have been assigned and approves the plan.

### 4.2.2 High Level Design Document (1 Month)

The Offeror shall deliver a high level design document describing the Offerors provided software and plans for how it may become the OpenIB stack. The design document shall include Linux kernel modules and plans for getting them into the 2.6.x and RHEL V3 Linux kernel. The Offeror shall schedule a design review face to face meeting with the University. This milestone is complete when Offeror survives a design review with the University and the UTR approves the design document.

### 4.2.3 OpenIB Kernel Module, HCA Driver, and Core Access Layer (3 Months)

The Offeror shall deliver a core OpenIB access layer, Linux kernel modules, and HCA device driver (for 2.6.x and RHEL V3). The core OpenIB access layer shall include a device driver framework and other functions that need to be provided to the device driver. The Offeror shall also deliver release notes and in-line documentation. This milestone is complete when the kernel modules and device driver are accepted into the mainline Linux kernel (www.kernel.org).

### 4.2.4 OpenIB Software Alpha Stack HPC Design Document (4.5 Months)

The Offeror shall deliver a design document for OpenIB Access Layer, connection management API, subnet management, and all HPC ULPs (MPI, IPoIB, SDP, and Sandia Portals) that enables the HPC capabilities in this SOW. This milestone is complete when Offeror reviews the design with Tri-Laboratory personnel and the UTR approves the design document.

### 4.2.5 Alpha Release of OpenIB Access Layer with HPC capabilities(6 Months)

The alpha code drop for the full (core and non-core) OpenIB access layer shall meet the HPC capabilities in this SOW. This release shall include API and CLI interfaces to CM, OpenSM, and SA. This milestone is complete when the OpenIB access layer is accepted into the mainline Linux kernel (www.kernel.org).

### 4.2.6 Alpha Release of OpenIB ULPs with HPC capabilities(9 Months)

The code drop for all HPC ULPs (MPI, SDP, IPoIB, and Sandia Portals) shall meet the HPC capabilities in this SOW. This milestone is complete when 1.) SDP and IPoIB, with HPC capabilities, are accepted into the mainline Linux kernel (www.kernel.org), 2.) Sandia Portals is ported to the OpenIB access layer, and 3.) Alpha release of MPI-2 that meets SOW requirements is provided.

### 4.2.7 OpenIB Stack Alpha Release HPC Test Plan (9 Months)

The Offeror, in collaboration with the Tri-Labs, shall provide a test plan for the Alpha release of the OpenIB stack with HPC capabilities. This milestone is complete when UTR approves the test plan.

### 4.2.8 OpenIB Software Stack Alpha Release with HPC capabilities(12 Months)

The Alpha code release for the OpenIB software stack shall meet the HPC capabilities contained in this SOW. The Alpha release shall include all devices drivers, kernel modules, access layer, OpenSM, CA, SA, HPC ULPs, and diagnostic and management tools. This milestone is complete when the Alpha Stack release meets the requirements of the Alpha stack test plan (section 4.2.7).

### 4.2.9 OpenIB Software Beta HPC Stack Design Document (12 Months)

The Offeror shall deliver a design document for the OpenIB Access Layer, connection management API, and subnet management second release. This release shall meet all of the functionality, performance and scalability and reliability requirements contained in this SOW. This milestone is complete when Offeror reviews the design with Tri-Laboratory personnel and the UTR approves the design document.

### 4.2.10 OpenIB Beta Stack with HPC capabilities Test Plan (15 Months)

The Offeror, in collaboration with the Tri-Labs, shall provide a test plan for the Beta release of the OpenIB stack with HPC capabilities. This milestone is complete when UTR approves the test plan.

### 4.2.11 OpenIB Software Beta Stack Release (15 months)

The Offeror shall deliver a Beta code release for the OpenIB software stack that meets the HPC capabilities contained in this SOW. The Beta release shall include device drivers, kernel modules, access layer, CM , OpenSM, SA, HPC ULPs, and diagnostic and management tools. This milestone is complete when the Beta Stack release meets the requirements of the Beta stack test plan (section 4.2.10) and when the stack is accepted into the mainline Linux kernel (www.kernel.org).

### 4.2.12 OpenIB Software Stack Final Release with HPC capabilities (18 Months)

The Offeror shall deliver an OpenIB Software Stack final release. This milestone is complete when the delivered OpenIB Software Stack meets all functionality, scalability, performance and reliability requirements of this SOW.

# 5 GLOSSARY

| Item | Description |
|---|---|
| 1X | An InfiniBand interface width. 1X defines an interface with 2 differential pairs, 1 Transmit, 1 Receive. Provides 2.5Gbit/s full-duplex connections. |
| 4X | An InfiniBand interface width. 4X defines an interface with 8 differential pairs (4 per direction), 4 Transmit, 4 Receive. Provides 10Gbit/s full-duplex connections. |
| 4X DDR | Double data rate InfiniBand interface. 4X defines an interface with 8 differential pairs (4 per direction). Providing 5.0 Gbits/s per differential pair for 20 Gbit/s full-duplex. |
| 12X | An InfiniBand interface width. 12X defines an interface with 24 differential pairs (12 per direction), 12 Transmit, 12 Receive. Provides 30Gbits/s full-duplex connections. |
| 12X DDR | Double data rate InfiniBand interface. 12X defines an interface with 24 differential pairs (4 per direction). Providing 5.0 Gbits/s per differential pair for 60 Gbits/s full-duplex. |
| API | Application programmer's interface. Syntax and semantics for invoking services from within an executing application. All APIs shall be available to both Fortran and C programs, although implementation issues (such as whether the Fortran routines are simply wrappers for calling C routines) are up to the supplier. |
| ASC | Advanced Simulation and Computing. Like its predecessor, ASCI, the ASC program is A U.S. Department of Energy program established to ensure confidence in the safety, performance, and reliability of its nuclear stockpile through better computer simulations. The ASC program is now active. |
| ASCI | Accelerated Strategic Computing Initiative. A U.S. Department of Energy program established to ensure confidence in the safety, performance, and reliability of its nuclear stockpile through better computer simulations. For the years 1996 through 2003, the ASCI program permitted the required degree of safety without nuclear testing. The follow-on program to ASCI is ASC. |
| BLAB | Aggregate Bi-directional Link Bandwidth (BLAB) is defined as the minimum of the aggregate memory bandwidth, aggregate bus bandwidth, or the sum bi-directional link peak user payload data bandwidth. |
| blocking operation | An operation that does not complete until the operation either succeeds or fails. For example, a blocking receive will not return until a message is received or until the channel is closed and no further messages can be received. |
| broadcast operation | A communication operation in which one processor sends (or broadcasts) a message to all other processors. |
| buffer | A portion of storage used to hold input or output data temporarily. |
| Clos | A network topology named after its inventor Charles Clos. |
| cluster | A set of SMPs connected via a scalable network technology. The network shall support high bandwidth, low latency message passing. It shall also support remote memory referencing. |

| Item | Description |
|---|---|
| CM (Connection Manager) | Service to establish, maintain, and status communication paths between remote peers. |
| collective communication | A communication operation that involves more than two processes or tasks. Broadcasts, reductions, and the MPI_Allreduce subroutine are all examples of collective communication operations. All tasks in a communicator must participate. |
| Communicator | An MPI object that describes the communication context and an associated group of processes. |
| Critical Path | The serial chain of dependencies that most limits forward progress. |
| DDR | Double data rate |
| Device Driver | Linux device driver to function the IB host channel adapter devices on a node. |
| DHCP | Dynamic Host Configuration Protocol – DHCP enables individual computers on an IP network to extract their configurations from a server (the 'DHCP server') or servers, in particular, servers that have no exact information about the individual computers until they request the information. DHCP is often used to reduce the work necessary to administer a large network (e.g. managing IP addresses). |
| Diagnostic & management tools | Commands and APIs to determine the hardware and software state of the IB configuration and manipulate that configuration. IB Access Layer - Low level API that exposes IB verbs, CM, SM and SA functionality in a vendor neutral manner. |
| Documentation | All provided software should be documented so that personnel unfamiliar with the IB stack structure may easily install and manage an IB cluster. Documentation should be provided so that personnel may make code enhancements and extensions to any software layer (driver, access layer, ULPs, diagnostics) of the IB software stack. |
| DOE | The U.S. Department of Energy. |
| DPCL | Dynamic Probe Class Library. |
| fairness | A policy in which tasks, threads, or processes must be allowed eventual access to a resource for which they are competing. For example, if multiple threads are simultaneously seeking a lock, no set of circumstances can cause any thread to wait indefinitely for access to the lock. |
| FT | Fault tolerance or Fault tolerant. |
| FY | Fiscal Year – The US government uses the fiscal year October 1 through September 30$^{th}$. For example, FY04 is October 1, 2003 through September 30, 2004. |
| GB: | Gigabyte. Gigabyte is a billion base 10 bytes. This is typically used in every context except for Random Access Memory size and is $10^9$ (or 1,000,000,000) bytes. |
| GiB: | GibiByte. GibiByte is a billion base 2 bytes. This is typically used in terms of Random Access Memory and is $2^{30}$ (or 1,073,741,824) bytes. |

| Item | Description |
|---|---|
| GPL | General Public License -- A legal software license arrangement developed by GNU to promote open software. The licenses for most software are designed to prevent users from sharing or changing it. By contrast, the GNU General Public License is intended to guarantee the freedom to share and change free software - to make sure the software is free for all its users. The GPL is designed to make sure that anyone can distribute copies of free software (and charge for this service if they wish); that they receive source code or can get it if they want; that they can change the software or use pieces of it in new free programs; and that they know they can do these things. The GPL forbids anyone to deny others these rights or to ask them to surrender the rights. These restrictions translate to certain responsibilities for those who distribute copies of the software or modify it. |
| GUI | Graphical user interface. A type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop. Within that scene are icons, which represent actual objects that the user can access and manipulate with a pointing device. |
| hot spot | A memory location or synchronization resource for which multiple processors compete excessively. This competition can cause a disproportionately large performance degradation when one processor that seeks the resource blocks, preventing many other processors from having it, thereby forcing them to become idle. |
| HCA | Host Channel Adapter. IBA expansion card that interfaces the IBA interconnect to the cluster node I/O subsystem. |
| HEC | High-end Computing |
| HT (HyperTransport) | HyperTransport. HyperTransport is an I/O link. With clock speeds of up to 1.4 GHz and Double Data Rate (DDR) signaling, HyperTransport technology provides an effective throughput of 2.8 gigatransfers per pin-pair on a 32-bit link. This results in a maximum aggregate throughput of 22.4 gigabytes per second, per link. (see http://www.hypertransport.org/tech_overview.html) |
| HPC ULPs | High Performance Computing Upper Layer Protocols include MPI, IPoIB, SDP, and Sandia Portals. |
| IBA | InfiniBand Architecture |
| IBTA | InfiniBand Trade Association (http://www.infinibandta.org/ibta/) |
| InfiniBand access layer | Includes the user-mode components for management services, SM query, connection management, and work request processing, and the kernel mode components for InfiniBand PnP, management services, resource management, connection management, work request processing, and user-level proxy agent |
| IPoIB | Internet Protocol over InfiniBand. IP specifies the format of packets, also called *datagrams*, and the addressing scheme. |
| iSCSI | iSCSI is Internet SCSI (Small Computer System Interface), an Internet Protocol (IP)-based storage networking standard for linking data storage facilities, developed by the Internet Engineering Task Force (IETF). By carrying SCSI commands over IP networks, iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. |

| Item | Description |
|---|---|
| iSER | iSCSI Extensions for RDMA |
| kDAPL | Kernel Direct Access Programming Library defines a single set of kernel-level APIs for all RDMA-capable Transports. The kDAPL mission is to define a Transport-independent and Platform standard set of APIs that exploits RDMA capabilities, such as those present in IB, VI, and iWARP. |
| Kernel modules | Changes to the Linux kernel needed to support the rest of the OpenIB software stack. |
| LAB | Aggregate Link Bandwidth (LAB) is defined as the minimum of the aggregate memory bandwidth, aggregate bus bandwidth, or the sum of uni-directional link peak user payload data bandwidth. |
| latency | The time interval between the instant at which an instruction control unit initiates a call for data transmission, and the instant at which the actual transfer of data (or receipt of data at the remote end) begins. Latency is related to the hardware characteristics of the system and to the different layers of software that are involved in initiating the task of packing and transmitting the data. |
| LVDS | Low voltage differential signaling. An electrical spec (EIA-644) used by InfiniBand. LVDS is designed with an output voltage swing of 350mV at better then 400Mbps into a 100 ohm load, across a distance of about 10 meters. |
| MPI | Message passing interface. An industry-standard message-passing protocol that typically uses a two-sided send-receive model to transfer messages between processes. |
| MTBF | A measurement of the expected reliability of the system or component. The MTBF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. |
| nonblocking operation | An operation, such as sending or receiving a message, that returns immediately whether or not the operation was completed. For example, a nonblocking receive will not wait until a message is sent, but a blocking receive will wait. A nonblocking receive will return a status value that indicates whether or not a message was received. |
| LANL | Los Alamos National Laboratory (http://www.lanl.gov/) |
| LC | Livermore Computing. The supercomputer center at Lawrence Livermore National Laboratory |
| LLNL | Lawrence Livermore National Laboratory (http://www.llnl.gov/) |
| M&IC | Multiprogrammatic and Institutional Computing. Organization responsible for providing unclassified computing to all programs at the University Lawrence Livermore National Laboratory. |
| MB | Megabyte. Megabyte is a million base $10^6$ bytes. This is typically used in every context except for Random Access Memory size and is 106 (or 1,000,000) bytes. |
| MiB: | MebiByte. MebiByte is a million base 2 bytes. This is typically used in terms of Random Access Memory and is 220 (or 1,048,576) bytes.. |
| MPI | Message Passing Interface. |
| MPI-2 | Extensions to the MPI standard. |

| Item | Description |
|---|---|
| MPI I/O | An MPI extension allowing for the manipulation of files on different file systems. |
| MR | Mandatory Requirement. Mandatory requirements are items that are essential to the University and reflect the minimum qualifications an Offeror must meet in order to have their proposal evaluated further for selection. |
| MTBF | A measurement of the expected reliability of the system or component. The MTBF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. |
| NIC | Network Interface Card. An expansion board you insert into a computer so the computer can be connected to a network. Most NICs are designed for a particular type of network, protocol, and media, although some can serve multiple networks. |
| NNSA | National Nuclear Security Agency. The arm of the DOE which is responsible for enhancing United States national security through the military application of nuclear energy. The NNSA maintains and enhances the safety, reliability, and performance of the United States nuclear weapons stockpile, including the ability to design, produce, and test, in order to meet national security requirements. The NNSA also supports United States leadership in science and technology. |
| OS | Operating System. |
| parallelism | The degree to which parts of a program may be concurrently executed. |
| PCI-Express | PCI Express is a dual-simplex, point-to-point serial differential low-voltage interconnect. Previously known as 3GIO and Arapahoe, PCI Express was announced in April, 2002. PCI-Express allows a bandwidth up to 500 MB/sec duplex for each link up to 8 GB/sec for sixteen lanes (x16). |
| PCI-X | A follow-on initiative to PCI (*Peripheral Component Interconnect)*. PCI-X allows a bandwidth up to 1GB/sec for 64 bit bus running at 133 MHz. [Note that we distinguish PCI-X and PCI-Express] |
| PERUSE | MPI Performance examination and revealing unexposed state extension specification – the specified API. |
| PMPI | Profiling interface for MPI specified by the MPI standard. |
| Portals (Sandia Portals) | Low-level API providing reliable and ordered communication for Lustre. (http://sourceforge.net/projects/sandiaportals) |
| POSIX | Portable Operating System Interface - A set of IEEE standards designed to provide application portability between Unix variants. IEEE 1003.1 defines a Unix-like operating system interface, IEEE 1003.2 defines the shell and utilities and IEEE 1003.4 defines real-time extensions. |
| QDR | Quad Data Rate |
| RC | Reliable Connection |
| RDMA | Remote Direct Memory Access (RDMA) capability allows a processes executing on one node of a cluster to be able to “directly” access (execute reads or writes against) the memory of processes within the same user job executing on a different node of the cluster. |

| Item | Description |
|---|---|
| reduction operation | An operation, usually mathematical, that reduces a collection of data by one or more dimensions. For example, the arithmetic SUM operation is a reduction operation which reduces an array to a scalar value. Other reduction operations include MAXVAL and MINVAL. |
| RHEL | RedHat Enterprise Linux |
| RMA | Remote Memory Access. A user-level communication protocol which provides ability for a task to access memory of another task by the use of put/get operations. |
| Scalability | Tested on 4,096 node physical fabrics and scaling properties simulated up to 16,384 nodes. |
| SDP | Sockets Direct Protocol - Sockets Direct Protocol (SDP) is an IBA specific protocol defined by the Software Working Group (SWG) of the IBA. The SDP specification maintains traditional sockets SOCK STREAM semantics as commonly implemented over TCP/IP, as well as support for byte-streaming over a message passing protocol, including kernel bypass data transfers and zero-copy data transfers. |
| SA (Subnet Administrator) | Service to store, track, and status the IB hardware configuration and routing information. |
| SM(Subnet Manager) | Software entity that discovers network topology, initializes the subnet, establishes routes, and provides regular subnet sweeps. |
| SNL | Sandia National Laboratories |
| SMP | Shared memory Multiprocessor. A set of CPUs sharing random access memory within the same memory address space. The CPUs are connected via a high speed, low latency mechanism to the set of hierarchical memory components. The memory hierarchy consists of at least processor registers, cache and memory. The cache shall also be hierarchical. If there are multiple caches, they shall be kept coherent automatically by the hardware. The main memory may be a Non-Uniform Memory Access (NUMA) architecture. The access mechanism to every memory element shall be the same from every processor. More specifically, all memory operations are done with load/store instructions issued by the CPU to move data to/from registers from/to the memory. A single SMP may be partitioned into one or more nodes. |
| SOW | Statement of Work |
| synchronization | The action of forcing certain points in the execution sequences of two or more asynchronous procedures to coincide in time. |
| Test harness and modules | Software to automatically test the functionality, performance, reliability, and robustness of the components of the OpenIB software stack. |
| TLP | Thread Level Parallelism. |
| thread | A single, separately dispatchable, unit of execution. There may be one or more threads in a process, and each thread is executed by the operating system concurrently. |
| Tri-lab | Refers to the three U.S. national security laboratories: Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratories. |

| Item | Description |
|---|---|
| TR-1 | Target Requirement, Priority 1. A highly desired characteristic or feature. |
| TR-2 | Target Requirement, Priority 2. A characteristic or feature that is a lower priority than corresponding TR-1 characteristics. |
| TR-3 | Target Requirement, Priority 3. A third-tier characteristic or feature that is a lower priority than corresponding TR-2 characteristics. TR-3 requirements are considered stretch goals that boost a moderately useful system, taken together as an aggregate of MR, TR-1, TR-2 and TR-3 requirements, into the highly useful category. |
| UD | Unreliable Datagram |
| uDAPL | User Direct Access Programming Library defines a single set of user-level APIs for all RDMA-capable Transports. The uDAPL mission is to define a Transport-independent and Platform standard set of APIs that exploits RDMA capabilities, such as those present in IB, VI, and RDDDP WG of IETF. |
| ULP (Upper Layer Protocols) | APIs for applications to perform IB communications operations. For this SOW, ULPs shall refer to MPI-2, IPoIB, SDP, and Sandia Portals. |
| URDMA | Unacknowledged, unreliable RDMA capability. |
| UTR | University Technical Representative |
| VAPI | InfiniBand Verbs Applications Programming Interface (VAPI). |